8. Interrupts, Traps, and Reset

The TMS34010 supports eight interrupts, including reset. Memory addresses >FFFF FC00 to >FFFF FFFF contain the 32 vector addresses used during interrupts, software traps and reset. Each vector is a 32-bit address that points to the beginning of the appropriate interrupt service routine.

This section includes the following topics:

Table 8-1 and Figure 8-1 (page 8-2) summarize the TMS34010 interrupts and their priorities. $\overline{\text{RESET}}$ has the highest priority, and the illegal opcode interrupt has the lowest. If two interrupts are requested at the same time, the highest priority interrupt is serviced first (assuming it is enabled). The reset and nonmaskable interrupt cannot be disabled.

Table 8-1. Interrupt Priorities

Int.	Priority	Internal/ External	Description and Source
Reset	1	I	[illegible] reset. Taken when the input signal at the [illegible] pin is asserted low.
NMI	2	I	Nonmaskable interrupt. Generated by a host processor.
HI	3	I	Host interrupt. Generated by a host processor.
DI	4	I	Display interrupt. Generated by the TMS34010.
WV	5	I	Window violation interrupt. Generated by the TMS34010.
INT1	6	E	External interrupts 1 and 2. Generated by external devices.
INT2	7	E	
ILLOP	8	I	Illegal opcode interrupt. Generated by the TMS34010 when an illegal opcode is encountered.

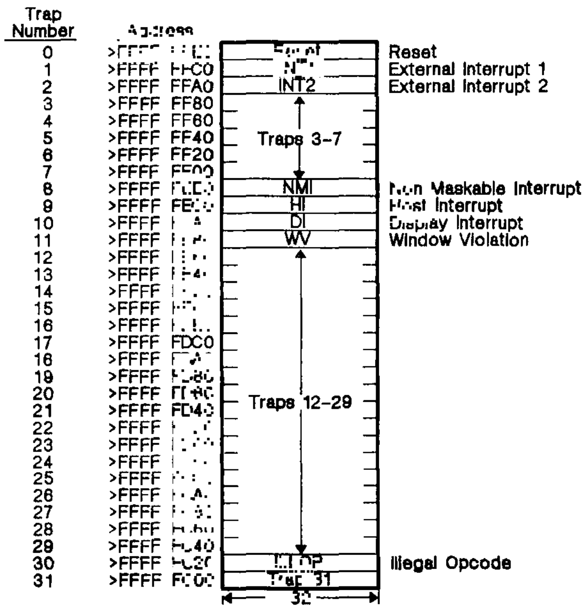


Figure 8-1. Vector Address Map

8.1 Interrupt Interface Registers

Two registers, a subset of the I/O registers discussed in Section 6, monitor *and* mask interrupt requests. These registers are summarized below; for more information, please refer to the register descriptions in Section 6.

The interrupt enable register, **INTENB**, contains the interrupt mask that selectively enables various interrupts. An interrupt is enabled when the status IE (global interrupt enable) bit and the appropriate bit in the INTENB register are *both* set to 1.

- *X1E* (bit 1) enables external interrupt 1.
- *X2E* (bit 2) enables external interrupt 2.
- *HIE* (bit 9) enables the host interrupt.
- *DIE* (bit 10) enables the display interrupt.
- *WVE* (bit 11) enables the window violation interrupt.

The interrupt pending register, **INTPEND**, indicates which interrupts are currently pending. When an interrupt is requested, the appropriate bit in the INTPEND register is set.

- *X1P* (bit 1) indicates that external interrupt 1 is pending.
- *X2P* (bit 2) indicates that external interrupt 2 is pending.
- *HIP* (bit 9) indicates that the host interrupt is pending.
- *DIP* (bit 10) indicates that the display interrupt is pending.
- *WVP* (bit 11) indicates that the window violation interrupt is pending.

8.2 External Interrupts

External interrupt requests are received via local interrupt pins $\overline{\text{LINT1}}$ and $\overline{\text{LINT2}}$. Each of the two external interrupt pins is dedicated to an individual interrupt, allowing two independent interrupt requests to be generated. (The pins are not encoded.) The local interrupt pins are level-sensitive, active-low inputs. Once an interrupt request has been initiated by driving an interrupt pin low, it must remain low until the GSP can respond to the interrupting device. This is necessary to ensure that the GSP detects the request. If the active level is maintained after returning from the interrupt service routine, however, the interrupt will be taken once again.

Signals input to the local interrupt pins are assumed to be asynchronous to the GSP local clocks, and are synchronized internally by the GSP before they are processed. If two external interrupt requests are active at the same time, INT1 will be serviced first. Table 8-2 shows the interrupt trap vectors for INT1 and INT2.

Table 8-2. External Interrupt Vectors

Name	Input Pin	Vector Address
INT1	$\overline{\text{LINT1}}$	>FFFF FFC0
INT2	$\overline{\text{LINT2}}$	>FFFF FFA0

8.3 Internal Interrupts

Several internal conditions are associated with specific interrupts. Table 8-3 summarizes these interrupts. If two internal interrupts are requested simultaneously, or if two or more internal interrupt requests are pending, the highest priority interrupt will be serviced first; NMI has the highest priority, followed by HI, DI, and WV. When internal *and* external interrupts are pending, the internal interrupts are serviced first (with the exception of the ILLOP interrupt).

Table 8-3. Interrupts Associated with Internal Events

Name	Function	Level	Vector Location	Description
NMI	Nonmaskable interrupt	8	>FFFF FEE0	The host processor sets the NMI bit in the HSTCTL register to a 1.
HI	Host interrupt	9	>FFFF FEC0	The host processor sets the INTIN bit in the HSTCTL register to a 1.
DI	Display interrupt	10	>FFFF FEA0	A particular horizontal line on the video display is being refreshed. The line number is specified in the DPYINT register.
WV	Window violation interrupt	11	>FFFF FE80	An attempt has been made to move a pixel to a destination location that lies inside or outside a specified window, depending on the selected windowing mode.
ILLOP	Illegal operand interrupt	30	>FFFF FC20	See Section 8.6.

The nonmaskable interrupt, or NMI, occurs when a host processor requests an interrupt by writing a 1 to the NMI bit in the HSTCTL register. This interrupt cannot be disabled, and will always occur as soon as possible following the request. The NMI will be delayed only for completion of an instruction already in progress, or until the next interruptible point of an interruptible instruction such as a PIXBLT is reached.

The NMI mode bit in the HSTCTL register determines whether or not context information is saved on the stack when a nonmaskable interrupt occurs:

- If NMIM = 0, the PC and ST are pushed on the stack before the interrupt is serviced.
- If NMIM = 1, nothing is saved on the stack before the interrupt is serviced.

The display interrupt (DI) is used to coordinate processing activity with the refreshing of particular areas of the display. The display interrupt request becomes active when a particular display line, specified in the DPYINT register, is output to the monitor screen. At the start of each horizontal blanking period, the VCOUNT register is compared to the DPYINT register. When the vertical count value in VCOUNT = DPYINT, a display interrupt request is generated. If enabled, the interrupt is taken.

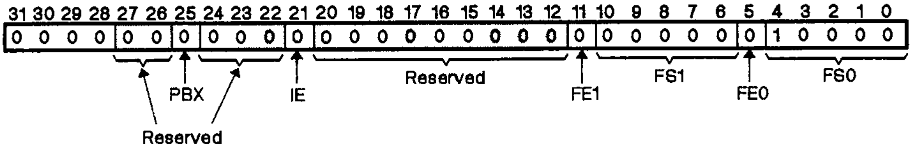
8.4 Interrupt Processing

An interrupt is said to be *pending* if it has been requested but has not yet been processed. If a pending interrupt is enabled, and no interrupt of higher priority is pending at the same time, the interrupt is accepted by the GSP at the end of the current instruction (or at the next interruptible point in the middle of a PIXBLT or FILL instruction). When the GSP takes an interrupt, it performs the following actions:

1) The GSP pushes the PC on the stack.

2) The GSP pushes the ST on the stack. PIXBLT and FILL instructions that are interrupted by external, host, and nonmaskable (if NMIM=0) interrupts set the PBX bit in the ST before pushing the ST.

3) The GSP modifies the contents of the ST as follows:



4) The GSP fetches the interrupt vector from external memory into the PC.

5) The GSP begins executing the instruction pointed to by the new PC value.

In step 5, the GSP resumes instruction execution at the entry point of the interrupt service routine. At the time the first instruction of the service routine begins execution, the new status register contents imply the following conditions:

- All interrupts are disabled (except NMI and reset)
- Field 0 is 16 bits long and is zero extended
- Field 1 is 32 bits long and is zero extended

The service routine can allow itself to be interrupted by loading a new interrupt-enable mask into the INTENB register and setting status bit IE to 1. The INTENB mask value is selected to determine which interrupts can interrupt the currently executing service routine. The service routine can also load new field sizes if values other than the defaults are required.

The last instruction in any interrupt service routine must be RETI (return from interrupt). Unlike the RETS (return from subroutine) instruction, which only pops the PC from the stack, RETI pops both the ST and PC. This restores the original state of the interrupted program so that execution can proceed from the point at which the interrupt occurred.

8.4.1 Interrupt Latency

An external interrupt, host interrupt request, or NMI request will be delayed by an amount of time that depends on the instruction in progress and on the local memory bus traffic at the time of the request.

The delay from an interrupt request to the time that the first instruction of the interrupt service routine begins execution is the sum of six potential sources of delay:

1) Interrupt request recognition
2) Screen-refresh cycle
3) DRAM-refresh cycle
4) Host-indirect cycle
5) Instruction interrupt
6) Interrupt context switch

In the best case, items 2 through 5 cause no delay. The minimum delay due to items 1 and 6 is 17 machine states.

- The **interrupt request recognition** delay is the time required for a request to be internally synchronized to the local clock. In the case of an external interrupt request, the delay is measured from the high-to-low transition of the $\overline{\text{INT}}$1 or $\overline{\text{INT}}$2 pin. In the case of a host interrupt or NMI request, the delay is measured from completion of the host's write to the INTIN or NMI pin.

- The **screen-refresh** and **DRAM-refresh cycles** are a potential source of delay, but in fact occur rarely and are unlikely to delay an interrupt.

- The likelihood of a delay caused by a **host-indirect cycle** is small in most instances, but this depends on the application. The delay due to a single host-indirect cycle is two machine states, assuming no wait states, but multiple host-indirect cycles occurring within a brief period of time could cause additional delays. Theoretically, a fast host processor could generate so many local memory cycles that the GSP would be prevented from servicing interrupts for an indefinite period.

- The **instruction interrupt** time refers to the time required for an instruction that was already executing at the time the interrupt request was received to either complete or to reach the next interruptible point in an instruction (such as a PIXBLT, FILL, or LINE).

- The **interrupt context switch** operation pushes the PC and ST onto the stack, and fetches the PC for the interrupt service routine from the appropriate vector in memory.

Table 8-4 shows the minimum and maximum times for each of the six operations listed. The interrupt latency is calculated as the sum of the numbers in the six rows. In the best case, the interrupt latency is only 17 machine states. The worst-case latency can be as high as 22 machine states plus the delays due to host-indirect cycles and instruction completion. Table 8-5 shows instruction interrupt times for some of the longer, noninterruptible instructions. Table 8-5 also shows the instruction completion time for a JRUC instruction that jumps to itself - the GSP may be executing this instruction if the software is simply waiting for an interrupt.

Table 8-4. Six Sources of Interrupt Delay

Operation	Latency (In States)	
	Min	Max
Interrupt recognition	1	2
Instruction interrupt	0	See Table 8-5
DRAM-refresh cycle	0	2 See Note 2
Screen-refresh cycle	0	2 See Note 2
Host-indirect cycle	0	See Note 1
Interrupt context switch	16	16

Notes:
1) The latency due to host-indirect cycles depends on both the hardware system and the application. Theoretically, a host processor could generate so many local memory cycles that the GSP could effectively be prevented from servicing interrupts. The delay due to a single host-indirect cycle is two machine states, assuming no wait states.
2) DRAM-refresh and screen-refresh cycle times assume no wait states.
3) Context switch time assumes that the SP is aligned to a word boundary; that is, the four LSBs of the SP are 0s. If the SP is not aligned, the delay is 28 states.

Table 8-5. Sample Instruction Completion Times

Instruction	Worst-Case Instruction Interrupt Time (In States)	
	SP Aligned	SP Not Aligned
DIVS A0,A2	43	43
MMFM SP,ALL	72	144
MMTM SP,ALL	73	169
Wait: JRUC wait	1	1

Notes:
1) The worst-case instruction interrupt time is equal to the instruction execution time less one machine state (except for PIXBLTs, FILLs, and LINE).
2) The SP-aligned case assumes that the SP is aligned to a word boundary in memory.

8.5 Traps

The TMS34010 supports 32 software traps, numbered 0 through 31. Software traps behave similarly to interrupts, except that they are initiated when the GSP executes a TRAP instruction. Unlike an interrupt, a software trap cannot be disabled.

When the GSP executes a TRAP instruction, it performs the same sequence of actions that it performs for interrupts. The TRAP 1 through TRAP 31 instructions cause the status register and the PC to be pushed onto the stack. TRAP 0 is similar to a hardware reset because it does not push the status register or PC onto the stack; it differs from a hardware reset because it does not cause the GSP's internal registers to be set to a known initial state. TRAP 8 is similar to an NMI interrupt, except that the NMIM (NMI mode) bit in the HSTCTLL register has no effect on instruction execution; the status register and PC are stacked unconditionally when TRAP 8 is executed.

A 32-bit vector address is associated with each software trap. To determine the vector address for a trap number *N*, where *N* = 0 through 31, subtract 32*N* from >FFFF FFE0. Figure 8-1 on page 8-2 shows the vector addresses for the software traps.

8.6 Illegal Opcode Interrupts

The GSP recognizes several reserved opcodes as illegal. When one of these opcodes is encountered in the instruction stream, the GSP will trap to vector number 30, located at memory address >FFFF FC20. An illegal opcode is similar in effect to a TRAP 30 instruction. The illegal opcode interrupt cannot be disabled. Table 8-6 lists ranges of illegal opcodes.

Table 8-6. Illegal Opcodes Ranges

>0000 through >00FF
>0200 through >02FF
>0400 through >04FF
>0800 through >08FF
>0A00 through >0AFF
>0C00 through >0CFF
>0E00 through >0EFF
>3400 through >37FF
>7000 through >7FFF
>9E00 through >9FFF
>BE00 through >BFFF
>D800 through >DEFF
>FE00 through >FFFF

8.7 Reset

Reset puts the TMS34010 into a known initial state. It is entered when the input signal at the RESET pin is asserted low. RESET must remain active low for a minimum of 40 local clock (LCLK1 and LCLK2) periods to ensure that the TMS34010 has sufficient time to establish its initial internal state.

While RESET remains asserted, all outputs are in a known state, no DRAM-refresh cycles take place, and no screen-refresh cycles are performed.

At the low-to-high transition of the RESET signal, the state of the HCS input determines whether the GSP will be halted or begin executing instructions. The GSP may be in one of two modes, host-present or self-bootstrap mode.

- **Host-Present Mode**

  If HCS is high at the end of reset, GSP instruction execution is halted and remains halted until the host clears the HLT (halt) bit in HSTCTL (host control register). Following reset, the eight RAS-only refresh cycles required to initialize the dynamic RAMs are performed automatically by the GSP memory control logic. As soon as the eight RAS-only cycles are completed, the host is allowed access to GSP memory. At this time, the GSP begins to automatically perform DRAM refresh cycles at regular intervals. The GSP remains halted until the host clears the HLT bit. Only then does the GSP fetch the level-0 vector address from location >FFFF FFE0 and begin executing its reset service routine.

- **Self-Bootstrap Mode**

  If HCS is low at the end of reset, the GSP first performs the eight RAS-only refresh cycles required to initialize the DRAMs. Immediately following the eight RAS-only cycles, the GSP fetches the level-0 vector address from location >FFFF FFE0, and begins executing its reset service routine.

Unlike other interrupts and software traps, reset does not save previous ST or PC values. This is because the value of the stack pointer just before a reset is generally not valid, and saving its value on the stack is unnecessary. A TRAP 0 instruction, which uses the same vector address as reset, similarly does not save the ST or PC values.

8.7.1 Asserting Reset

A reset is initiated by asserting the RESET input pin at its active-low level. To reset the GSP at power up, RESET must remain active low for a minimum of 40 local clock periods after power levels have become stable. At times other than power up, the GSP is also reset by holding RESET low for a minimum of 40 clock periods. The 40-clock interval is required to bring GSP internal circuitry to a known initial state. While RESET remains asserted, the output and bidirectional signals are driven to a known state.

The GSP drives its RAS signal inactive high as long as RESET remains low. The specifications for certain DRAM and VRAM devices, including the TMS4161, TMS4164 and TMS4464 devices, require that the RAS signal be driven inactive-high for 100 microseconds during system reset. Holding RESET low for

150 microseconds will cause the RAS signal to remain high for the 100 microseconds required to bring the memory devices to their initial states. DRAMs such as the TMS4256 specify an initial RAS high time of 200 microseconds, requiring that RESET be held low for 250 microseconds. In general, holding RESET low for *t* microseconds ensures that RAS remains high initially for *t* - 50 microseconds.

8.7.2 Suspension of DRAM-Refresh Cycles During Reset

An active-low level at the RESET pin is considered to be a power-up condition, and DRAM refresh is not performed until RESET goes inactive high. Consequently, the previous contents of the local memory may not be valid after a reset.

8.7.3 Initial State Following Reset

While the RESET pin is asserted low, the GSP's output and bidirectional pins are forced to the states listed in Table 8-7.

Table 8-7. State of Pins During a Reset

Outputs Driven To High level	Outputs Driven To Low Level	Bidirectional Pins Driven to High Impedance
DDOUT	BLANK	HSYNC
[illegible]		VSYNC
[illegible]		HD0-HD15
LAL		LAD0-LAD15
TR/QE		
RAS		
CAS		
W		
HINT		

Immediately following reset, all I/O registers are cleared (set to >0000), with the possible exception of the HLT bit in the HSTCTL register. The HLT bit is set to 1 if HCS is high just before the low-to-high transition of RESET.

Just before execution of the first instruction in the reset routine, the TMS34010's internal registers are in the following state:

- General-purpose register files A and B are uninitialized.
- The ST is set to >0000 0010.
- The PC contains the 32-bit vector fetched from memory address >FFFF FFE0.

The instruction cache is in the following state at this time:

- The SSA (segment start address) registers are uninitialized.
- The LRU (least recently used) stack is set to the initial sequence 0,1,2,3, where 0 occupies the most-recently-used position, and 3 occupies the least-recently-used position.
- All P (present) flags are cleared to 0s.

8.7.4 Activity Following Reset

Immediately following the low-to-high transition of RESET, the GSP performs a series of eight RAS-only memory cycles to bring the DRAMs and VRAMs to their initial operating states. These cycles are completed before any accesses of the GSP's memory (by either the GSP or host processor) are allowed to occur. If the host processor attempts to access the GSP memory indirectly before the eight RAS-only cycles have completed, it will receive a not-ready signal from the GSP until the cycles have completed. The eight RAS-only cycles occur regardless of the initial value to which the HLT bit in the HSTCTL register is set.

Each of the eight RAS-only cycles is a standard DRAM-refresh cycle. The RF bus status signal output with the row address is active low. The row address is all 0s.

Following the eight RAS-only cycles, the GSP automatically begins to initiate a new DRAM-refresh cycle every 32 GSP local clock cycles. The first DRAM refresh cycle begins approximately 32 local clock periods after the end of reset. A DRAM-refresh cycle will continue to be initiated every 32 GSP clock cycles until the DRAM-refresh rate is changed by the GSP or host processor.

The GSP is configured by means of an external signal input on the HCS pin to either:

- Begin executing instructions immediately after reset is completed (self-bootstrap mode)
  or
- Halt until the host processor instructs it to begin executing (host-present mode)

8.7.4.1 Self-Bootstrap Mode

In self-bootstrap mode, the GSP begins executing instructions immediately following reset. This mode is typically used in a system in which the reset vector and reset service routine are contained in nonvolatile memory, such as a bootstrap ROM. This type of system does not necessarily require a host processor, and the GSP may be responsible for performing host processor functions for the system.

The GSP is configured in self-bootstrap mode when the HCS pin is low just before the low-to-high transition of RESET. The low HCS level forces the HLT bit to 0. Immediately following the end of reset and the eight RAS-only cycles, the GSP fetches the level-0 vector address and begins executing the reset interrupt routine.

At the low-to-high transition of RESET, the HCS input is internally delayed before being checked to determine how to set the HLT bit. In a system without a host processor, for instance, this permits the HCS and RESET pins to be tied together, eliminating the need for additional external logic.

Transitions of the HCS and RESET signals are assumed to be asynchronous with respect to the GSP local clock. HCS and RESET are internally synchronized to the local clock by being held in latches for at least one clock period before being used by the GSP. The delay through the synchronizer latch is from one to two local clock periods, depending on the phase of the signal transitions relative to the clock. To permit the HCS and RESET pins to be wired

together, GSP on-chip logic delays the $\overline{\text{HCS}}$ low-to-high transition to ensure that it is detected **after** the $\overline{\text{RESET}}$ low-to-high transition. The level of the delayed $\overline{\text{HCS}}$ signal at the time the low-to-high $\overline{\text{RESET}}$ transition is detected determines the setting of the HLT bit.

8.7.4.2 Host-Present Mode

Host-present mode assumes that a host processor is connected to the GSP's host interface pins. In this mode, the GSP local memory can be composed entirely of RAM (no ROM). Following reset, the host processor must download the initial program code, interrupt vectors, and so on, before allowing the GSP to begin executing instructions.

The GSP is configured in host-present mode as follows. On the trailing edge of $\overline{\text{RESET}}$, the $\overline{\text{HCS}}$ (host interface chip select) input is sampled. If the $\overline{\text{HCS}}$ pin is inactive high, internal logic forces the HLT (halt) bit to a 1. In this fashion, the GSP is automatically halted following reset, and will not begin execution of its reset service routine until the host processor loads a 0 to HLT. In the meantime, the host processor is able to load the memory and I/O registers with the appropriate initial values before the GSP begins executing instructions. This may include writing the reset vector and reset service routine into the GSP's memory, for example.

No additional external logic is required to force $\overline{\text{HCS}}$ high before the low-to-high transition of $\overline{\text{RESET}}$. The simple external decode logic typically used will drive the $\overline{\text{HCS}}$ input active low only when one of the GSP's host interface registers is addressed by the host processor. Assuming that the host processor is not actively chip-selecting the GSP at the end of reset, $\overline{\text{HCS}}$ is high.